PEMBANGUNAN GAME *PC* BERBASIS 3D DENGAN TEKNOLOGI XNA DAN WIIMOTE

Michael Yoseph Ricky; Tunjung Agung Wijaya

Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Bina Nusantara
Jln. K.H. Syahdan No. 9, Palmerah, Jakarta Barat 11480
laboratory@binus.ac.id

ABSTRACT

The article explains about building a 3D based computer with XNA game engine and WiiMote technology as the controller or in other words, the input device. XNA is a game engine developed by Microsoft for building 2D or 3D game purposes in Xbox console. This engine licenses are free, but not for commercial purposes. WiiMote is Nintendo Wii's remote control. The remote can be connected to PC by bluetooth connection.

Keywords : *Game 3D,XNA, WiiMote Library*

ABSTRAK

Artikel membahas pembangunan sebuah game komputer yang berbasis 3D dengan menggunakan game engine XNA dan teknologi WiiMote sebagai alat pengendali atau dengan kata lain alat inputnya. XNA adalah sebuah game engine dari microsoft yang dipakai untuk membuat game 3D atau 2D pada konsol XBOX. Lisensi engine ini gratis, tapi tidak untuk kepentingan komersial. WiiMote adalah remote control untuk konsol game Nintendo Wii. Remote tersebut bisa dihubungkan ke komputer melalui bluetooth.

Kata kunci : *Game 3D,XNA,WiiMote Library*

PENDAHULUAN

Perkembangan teknologi dewasa ini sangatlah pesat. Mulai dari teknologi komunikasi, komputer, bahkan untuk membuat video game. Perkembangan video game bermula dari yang hanya bergambar kotak – kotak yang berpantulan, kemudian mulai bergambar karakter – karatker manusia 2 dimensi (2D), hingga sekarang ini bergambar 3 dimensi (3D). Untuk membuat game 3D, sudah bukan sesuatu yang baru dan sulit. Banyak game engine yang membantu untuk memgembangkan game 3D. Microsoft juga mengeluarkan game engine untuk membangun game pada konsol XBOX. Game engine tersebut adalah XNA. Selain untuk game 3D, XNA juga mendukung grafik 2D.

Perkembangan video game tidak hanya seputar penyajian gambarnya, tapi juga dengan perkembangan media penyimpanan data, kecepatan dan kekuatan konsol, dan juga perkembangan *controller*-nya. Nintendo Wii misalnya, *controller* yang digunakan adalah WiiMote (Wii Remote). Controller ini tidak menggunakan kabel. Komunikasi yang dilakukan antara WiiMote dengan konsolnya dengan menggunakan sinyal bluetooth. Fitur yang disajikan tidak hanya itu saja. WiiMote juga dilengkapi dengan accelerometer, yaitu sebuah alat yang bisa mengukur percepatan gerakan, kamera infra merah beresolusi 1024 x 768. Dengan kemajuan teknologi yang dijelaskan di atas, kami mencoba untuk menggabungkan dua teknologi tersebut. Kami membuat game 3D dengan game engine XNA dengan Nintendo WiiMote sebagai *controller*-nya.

PEMBAHASAN

Seputar Game Yang Dibuat

Game yang dibuat ini menceritakan tentang seorang anak kepala suku yang belajar berburu. Karena masih pada zaman batu, maka alat berburu yang digunakan pada saat itu adalah batu. Cara bermain game ini adalah pemain melempar batu ke arah sasaran yang bermacam – macam. Sasaran lemparan ada yang bergerak dan ada yang diam. Tiap sasaran memiliki poin jika dikenai. Waktu dan posisi kemunculan sasaran berbeda – beda. Pada game ini wiimote digunakan sebagai controller utama. Yaitu untuk melempar batu. Yang digunakan bukan sekedar tombol ditekan lalu batu terlempar, tapi accelerometer pada wiimote akan menentukan kekuatan lemparan batu.

Pengenalan XNA

Seperti yang sudah disebutkan sebelumnya, XNA adalah game engine yang dikeluarkan Microsoft untuk membangun game 2D atau pun 3D padakonsol XBOX.XNA bukanlah sebuahsingkatan, jadi tidak memiliki kepanjangan atau memiliki arti khusus. XNA menggunakan bahasa pemrograman C#. Karena itu, pemrograman XNA adalah pemrograman berbasis objek. Versi XNA yang terbaru adalah 3.1, dan versi tersebut yang kami digunakan. Untuk menggunakan framework atau pustaka XNA, yang dibutuhkan adalah Microsoft Visual Studio 2008 atau Visual C# 2008 Express Edition. IDE yang mendukung XNA saat ini masih dua IDE tersebut di atas. Untuk Microsoft Visual Studio 2010 masih belum bisa digunakan.

WiiMote

Pada November 2006, Nintendo meluncurkan konsol video game kelimanya , Nintendo Wii. Konsol video game sebelumnya Nintendo Gamecube, kurang bersaing dengan baik dengan Microsoft dan Sony pada saat itu. Akan tetapi, Nintendo Wii bisa menjadi penguasa pasaran setelah 1 tahun.

Aspek yang paling menarik dari Nintendo Wii adalah cara pemain memainkan game yang ada pada Nintendo Wii. Kebanyakan game yang ada menuntut para pemainnya untuk lebih bergerak untuk bisa berinteraksi dengan game. Tidak seperti konsol game yang lainnya yang cukup dengan menekan tombol, atau kombinasi tombolnya untuk melakukan suatu aksi. Pada game memancing misalnya, pemain tidak sekedar menekan tombol untuk menggerakkan karakter agar melempar pancing, tapi pemain harus mengayunkan wiimote seolah mengayunkan pancing yang sebenarnya. Kekuatan ayunan juga ditentukan oleh percepatan gerakan wiimote yang diayunkan pemain.

Fitur Wiimote yang digunakan

Wiimote memiliki banyak fitur yang bisa digunakan, tapi tidak semuanya digunakan dalam pembangunan game ini.Berikut fitur yang akan digunakan:

1. Basic Button
   Ada 12 tombol pada wiimote. 4 untuk tombol arah, tombol A, B, 1, 2, home, plus, minus dan power.
2. Vibration
   Wiimote dilengkapi juga vibration motor yang berguna menggetarkan wiimote. Fitur ini biasa digunakan untuk memberi efek getar jika misalnya ada benturan atau ledakan pada gmae sehingga pemain merasakan sensasi seolah ikut terkena dampak getarannya.
3. 3-axis accelerometer.
   Fitur ini adalah salah satu fitur yang membedakan dengan controller game console yang lain. Accelerometer pada wiimote memiliki 3 arah, arah X, Y, dan Z.

Wiimote Library

Agar kita bisa mengakses dan memberi informasi wiimote pada program yang kita bangun, kita memerlukan pustaka tambahanyaitu wiimotelib. Untuk mendapatkan pustaka ini, bisa diunduh dari http://wiimotelib.codeplex.com/.

Pustaka ini dibuat oleh Brian Peek dengan basis bahasa C# dan VB. Pustaka ini berupa file .dll yang bisa kita tambahkan pada project visual C# atau VB.net. Versi yang sedang dibangun oleh Brian Peek adalah 2.0, tapi yang akan dipakai pada pembuatan game ini adalah versi 1.7 karena versi 2.0 belum dirilis oleh Brian Peek.

Dasar Pemrograman XNA

Pada saat pembuatan game dengan XNA ada konsep – konsep dasar yang perlu diketahui. Setiap class utama yang akan dijalankanpada fungsi main haruslah turunan dari class Game pada XNA.Framework. Class Game mempunya beberapa method utama yang harus di-*overrride*pada class turunannya. Method tersebut adalah

1. void Initializes()
   Method ini dijalankan pertama kali sebelum method LoadContent, setelah constructor dijalankan. Method ini digunakan menginisialisasi nilai awal variable, atau inisialisasi logika game. Penempatan inisialisasi tidak harus di method ini, bisa langsung pada constructor.
2. void LoadContent()
   Method ini berisi code yang berguna me-load semua aset yang akan digunakan pada game. Aset – aset tersebut biasanya berupa suara, gambar, atau model 3D.
3. void UnloadContent()
   Method ini kebalikan dari loadContent, yaitu untuk menghapus data – data aset yang tadi dipakai selama program berjalan.
4. void Update()

Method ini dijalankan berulang selama program berjalan. Code pengambilan input dari user baik dari keyboard, mouse, atau pun gamepad, logika pergerakan object game, dan perubahan attribute ditaruh di sini.

5. void Draw()
   Method ini juga dijalankan berulang selama program berjalan, hanya saja setelah method update dijalankan. Method ini berguna untuk menggambar semua asset yang digunakan pada game baik 2D maupun 3D.

Selain 5 method di atas, sebuah class utama harus memiliki objek dari class GraphicDeviceManager dan SpriteBatch. GraphicDeviceManager adalah class yang bertugas mengatur atribut yang berhubungan dengan grafik pada game. Mulai dari ukuran resolusi layar, sampai ke masalah dukungan transparansi. SpriteBatch adalah class yang bertugas untuk menggambar 2D dantext pada layar monitor. Karena untuk 2D, maka hanya 2 sumbu koordinat saja yang dibutuhkan oleh SpriteBatch.

Menggambar Objek 3D

Pada game 3D, untuk menghasilkan gambar sebuah objek tidak semudah pada game 2D. Banyak hal yang berbeda. Di sini dibutuhkan model 3D, texture, cahaya (lights), dan kamera. Sistem koordinatnya juga berbeda. Model dapat dibuat dengan software 3D seperti Autodesk 3dsMax, MilkShape, Maya, dan lain sebagainya. Format yang bisa diterima oleh XNA adalah

Untuk mempermudah coding kedepannya, perlu dibuat sebuah class yang merepresentasikan objek pada game. Kita namakan class ini GameObject. Berikut ini atribut dan method yang perlu ada :

Atribut

1. Model model
2. Vector3 position
3. Vector3 rotation
4. Vector3 velocity
5. Boolean isAlive
6. String name
7. Float alpha
8. Float scale
9. Float lifetime
10. Boolean isCollided

Method:

1. Void loadContent
2. Void draw
3. Void move

Selain objek, diperlukan juga camera. Pada konsep 3D, camera digunakan untuk menentukan sudut pandang pengambilan gambar. Berbeda dengan game 2D yang bisa langsung menggunakan koordinat X dan Y layar, game 3D tidak bisa begitu. Koordinat X, Y, dan Z harus diubah dulu menjadi 2D layar. Oleh karena itu camera dibutuhkan, sebagai penentu perhitungan pengubahan koordinat tadi. Camera juga perlu dibuatkan classnya. Berikut ini attribut dan method yang perlukan.

Atribut:

1. Matrix view
2. Matrix projection
3. Vector3 position
4. Vector3 rotation
5. Vector3 targetPos
6. Vector3 UpVector
7. Float viewDistance

Method:
1. Void update()
2. Void update(Vector3 targetpos)

Dengan dua class ini, kita sudah bisa mulai menggambar objek 3D. Setelah membuat objek dari kelas GameObject, kita harus menginisialisasi modelnya.

```
Object.Model = Content.Load<Model>("Model\\batu");
```

Code di atas ditaruh di method LoadContent. Setelah menginisialisasi model, atur juga posisi, skala, rotasi, dan kecepatan jika ingin bergerak.Objek dari camera juga perlu dibuat. Berbeda dengan GameObjek, camera tidak memiliki model, karena tidak ada yang perlu digambar, hanya koordinatnya saja yang diperlukan.

Berikut code untuk menggambar sebuah objek.

```
public virtual void draw(Camera camera)
{
    foreach (ModelMesh mesh in model.Meshes)
    {
        foreach (BasicEffect effect in mesh.Effects)
        {
            effect.EnableDefaultLighting();
            effect.PreferPerPixelLighting = true;
            effect.Alpha = alpha;
            effect.AmbientLightColor = new Vector3(0.15f, 0.15f, 0.15f);
            effect.DiffuseColor = new Vector3(1.0f, 1.0f, 1.0f);
            effect.World = Matrix.CreateFromYawPitchRoll(
                rotation.Y,
                rotation.X,
                rotation.Z) *
                Matrix.CreateScale(scale) * Matrix.CreateTranslation(position);
            effect.Projection = camera.projection;
            effect.View = camera.view;
            effect.CommitChanges();

        }
        mesh.Draw();
    }
}
```

Effect yang digunakan untuk menggambar objek 3D pada code di atas adalah BasicEffect. Class tersebut sudah ada pada framework XNA. Tapi jika ingin, kita juga bisa membuatnya sendiri. Pada class BasicEffect dapat kita atur attribut dasar seperti alpha, warna diffuse, dan warna ambient.

Pada code di ataslah camera digunakan. Seperti yang terlihat, atribut view dan projection dari camera digunakan. Kedua atribut tersebut adalah matrix yang didapat dari hasil perhitungan pada method update pada class camera. Matrix tadi termasuk yang menentukan proses pengubahan koordinat 3D menjadi 2D pada layar. Pada class camera, ada dua buah fungsi update. Pertama, method update untuk membuat matrix view dan projection yang target posisinya berdasarkan perhitungan posisi,rotasi, dan jarak pandang camera. Kedua, method update yang target posisinya langsung ditentukan bukan berdasarkan perthitungan seperti pada method yang pertama.

```
public virtual void UpdateMatrices()
{
    rotationMatrix = Matrix.CreateRotationX(rotation.X) * Matrix.CreateRotationY(rotation.Y);
    targetPos = position + Vector3.Transform(new Vector3(0, 0, 1), rotationMatrix);
    upVector = Vector3.Transform(new Vector3(0, 1, 0), rotationMatrix);

    direction = Vector3.Normalize(targetPos - position);

    view = Matrix.CreateLookAt(position, targetPos, upVector);
    projection = Matrix.CreatePerspectiveFieldOfView(fov, aspectRatio, NearPlane, FarPlane);
}

public virtual void UpdateMatrices(Vector3 targetPos) {
    this.targetPos = targetPos;
    upVector = Vector3.Up;
    view = Matrix.CreateLookAt(position, targetPos, upVector);
    projection = Matrix.CreatePerspectiveFieldOfView(fov, aspectRatio, NearPlane, FarPlane);

    direction = view.Forward;
    rotationMatrix = Matrix.CreateRotationY(direction.Y);
    rotation.Y = (float)(Math.Atan2(direction.Z, direction.X) - atan2Y);
}
```

Menggerakkan Object 3D

Ada dua macam pergerakan yang bisa dibuat pada game 3D. Pergerakan yang pertama adalah paling mudah karena hanya pergerakan biasa yaitu pergerakan yang hanya bergantung pada koordinat dunia 3D.

Pergerakan yang orientasinya hanya koordinat dunia 3D konsepnya seperti pada 2D, yaitu hanya penambahan atau pengurangan posisi dengan kecepatannya. Untuk membuat pergerakan seperti cukup memberi nilai pada atribut Vector3 Velocity pada objek yang digerakkan. Setelah itu panggil method move objek tersebut di method Update pada class utama.

Jenis pergerakan kedua adalah pergerakan yang bukan hanya bergantung pada koordinat dunia 3D, tapi juga rotasi dan posisi camera. Pergerakan ini yang agak sulit. Kita harus menghitung matrix rotasi objek tersebut terlebih dahulu kemudian dinormalisasi menjadi Vector3.

Pemberian nilai atribut Velocity pada pergerakan ini tidak semudah pergerakan pertama. Nilai yang dimasukkan berdasarkan perhitungan posisi dan rotasi camera. Perhitungannya dapat dilihat pada code berikut.

```
public Vector3 GetMuzzleVelocity(GameObject gameObject, Camera cam, float power)
{
    Vector3 rotation = Vector3.Zero;
    if (gameObject != null)
    {
        rotation = gameObject.rotation;
    }
    else if (cam != null)
    {
        rotation = cam.rotation;
    }

    Matrix rotationMatrix =
        Matrix.CreateFromYawPitchRoll(rotation.Y, rotation.X, 0);
    return Vector3.Normalize(
            Vector3.Transform(Vector3.Forward, rotationMatrix)) * power;
}
```

Pada code di atas, perhitungan kecepatan juga bisa berdasarkan gameobject lainnya. Akan tetapi tidak bisa keduanya. Oleh karena pada code di atas divalidasi mana yang akan digunakan.

Mengambil Input WiiMote

Seperti yang dijelaskan sebelumnya, WiiMote digunakan untuk mengambil inputan dari pemain. Aksi melempar batu pada game ini dikendalikan dengan accelerometer pada WiiMote.

Nilai Sumbu Y pada accelerometer WiiMote ketika WiiMote bergerak maju mundur. Nilai sumbu Z pada accelerometer WiiMote ketika WiiMote bergerak atas bawah.Nilai sumbu X pada accelerometer WiiMote ketika WiiMote diputar ke kiri atau ke kanan.

Sumbu akselerasi pada WiiMote yang digunakan untuk aksi melempar batu hanya sumbu Y dan Z. Sumbu Y untuk mempengaruhi percepatan ke atas dan Z untuk mempengaruhi daya lempar ke depan.

Untuk mengambil nilai – nilai tersebut dalam code project program, yang perlu dilakukan adalah :

1. Menghubungkan WiiMote ke komputer dengan BlueTooth.
2. Tambahkan pustaka WiiMoteLib.
3. Buat objek dari class WiiMote.

```
WL.Wiimote wm = new WL.Wiimote();
```

4. Panggil fungsi connect() dari objek WiiMote tersebut pada constructor class utama.

```
try
{
    wm.Connect();
    wm.SetReportType(WL.InputReport.IRExtensionAccel,
        WL.IRSensitivity.Maximum, true);
    wm.SetLEDs(1);
    isWiimoteDetected = true;
}
catch (Exception)
{
    String pesan = "There is no wiiMote detected." + Environment.NewLine +
        " Please make sure the wiiMote connected properly";
    Forms.MessageBox.Show(pesan,"Error",
        Forms.MessageBoxButtons.OK,
        Forms.MessageBoxIcon.Error);
    this.Exit();
}
```

5. Untuk mengambil informasi wiimote, buat Objek dari kelas WiiMoteState.

```
WL.WiimoteState ws;
```

6. Buat code pengambilan informasi pada method update() dilakukan berulang kali.

```
protected override void Update(GameTime gameTime)
{
    KeyboardState keyState = Keyboard.GetState();
    if (isWiimoteDetected)
        ws = wm.WiimoteState;
    else
        ws = new WL.WiimoteState();
```

Kini WiiMote sudah terhubung dan informasi dari WiiMote sudah bisa diambil. Selanjutnya kita ambil informasi accelerometer dan tombol B sebagai trigger untuk melakukan aksi melempar batu.

Pada method update, diambil dulu button apa yang ditekan pada wiimote. Jika tombol B, maka ambil nilai akselerasi Y dan Z. Jika tombol dilepas, maka jalankan fungsi ThrowStone untuk melakukan aksi melempar batu pada game.

```
if (ws.ButtonState.B)
{
    power = Math.Abs(ws.AccelState.Values.Z * 2);
    upPower = ws.AccelState.Values.Y;
    isPressed = true;
    wm.SetRumble(false);
}
```

Variabel power adalah untuk menyimpan nilai daya lempar kedepan. Variabel upPower adalah untuk menyimpan nilai percepatan ke atas.

Nilai dari power didapat dari nilai Akselerasi sumbu Z yang diabsolutkan. Nilai tersebut diabsolutkan untuk mencegah nilai negatif. Karena jika nilai yang diterima negatif, maka akan membuat batu yang dilempar mengarah ke belakang, bukan ke depan. Variable isPressed digunakan untuk mengecek apakah kondisi sekarang sedang masih menahan tombol atau tidak.

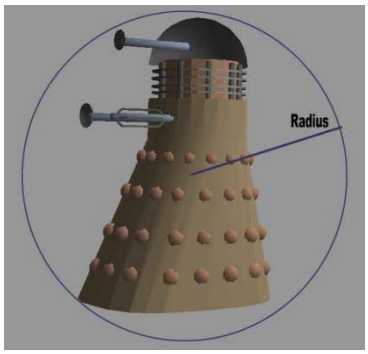
```
if (isPressed && !ws.ButtonState.B)
{
    ThrowStone(1, power, upPower);
    isPressed = false;
}
```

Code di atas adalah ketika tombol B sudah tidak ditekan lagi.

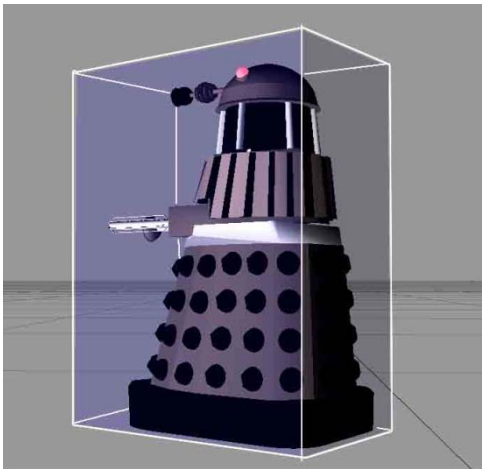
Collision Detection

Setelah mengatur input untuk melakukan aksi melempar batu, hal selanjutnya ada pendeteksian benturan objek. Hal ini disebut collision detection. Pada XNA, untuk membuat collision detection diperlukan yang namanya bounding object. Bounding object inilah yang digunakan untuk mengecek benturan antar objek yang berbeda. Ada 3 macam bounding object pada XNA, yaitu bounding frustum, bounding box, dan bounding sphere. Bounding frustum biasanya digunakan untuk membuat bentuk bangun yang terbentuk dari view dan proyeksi kamera. Bounding Box dan Bounding Sphere digunakan untuk mewakili bentuk objek 3D yang akan dicek benturannya.

Bounding Box lebih sulit membuatnya dari pada Bounding Sphere, akan tetapi lebih bisa diandalkan. Pada XNA, tidak sediakan fasilitas langsung untuk membuat Bounding Box, harus membuatnya sendiri. Caranya dengan mengambil verteks (titik) terjauh pada objek yang akan dibuatkan Bounding Boxnya. Sedangkan pada Bounding Sphere, cukup dengan mengambil data bounding sphere dari objek kemudian mengatur titik pusat dan radiusnya saja.



Sumber: (Ditchburn, N.D.)
Gambar 1 gambaran Bounding Sphere pada objek 3D.



Sumber : (Ditchburn, N.D.)
Gambar 2 gambaran Bounding Box pada objek 3D

Seperti yang terlihat pada gambar, bounding box lebih pas dari pada bounding sphere.Hal ini dikarenakan ukuran box lebih mendekati ukuran objek.

Karena yang mudah menggunakan Bounding Sphere, maka itu yang digunakan. Seperti yang dijelaskan sebelumnya, atribut Bounding Sphere yang perlu diberi nilai hanya titik pusat dan radiusnya saja. Nilai tersebut diambil dari posisi dan skala objek yang akan dicek benturannya. Berikut penggalan code-nya.

```
bool testCollision(GameObject a, GameObject b)
{
    BoundingSphere aSphere = a.model.Meshes[0].BoundingSphere;
    aSphere.Center = a.position;
    aSphere.Radius *= a.scale ;

    BoundingSphere bSphere = b.model.Meshes[0].BoundingSphere;
    bSphere.Center = b.position ;
    bSphere.Radius *= b.scale;
    return aSphere.Intersects(bSphere);

}
```

Untuk mengecek benturan cukup dengan method Intersects yang ada pada objek Bounding Sphere. Method ini menerima parameter berupa objek bounding lainnya dan mengembalikan data boolean.

PENUTUP

Selanjutnya yang harus dikerjakan adalah meneruskan pembuatan game sesuai dengan desain game yang sudah ditentukan. Mulai dari cerita, sistem penilaian, desain tingkat kesulitan tiap levelnya, dan fitur – fitur lainnya.

DAFTAR PUSTAKA

Ditchburn, K. (N.D.) *XNA Model Collisions*, diunduh dari http://www.toymaker.info/Games/XNA/html/xna_model_collisions.html.

Lee, J. (2008) *Hacking the Wii Remote*, TED, http://www.youtube.com/watch?v=QgKCrGvShZs

Nintendo (2008). *Consolidated Financial Highlights 2008*. Diunduh dari http://www.nintendo.co.jp/ir/pdf/2008/080124e.pdf.

Peek, B. (2007). *Managed Library for Nintendo's Wiimote*, Channel 9 MSDN. Diunduh dari http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx